CMPSC-265
Data Structures and Algorithms

Zaihan Yang
zyang13@suffolk.edu

Department of Math and Computer Science
Suffolk University

Fall 2019

Notice

- HW4 posted. Will be due on this Sunday midnight.
- Will finish grading till HW3 this week.
- You will have your first quiz this Wednesday.

Recap

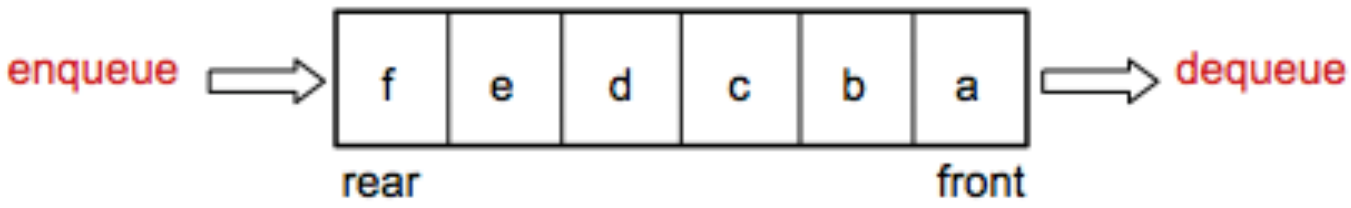
- Sorting algorithm
- Implementing Comparable and Comparator interface to sort on objects
- The Stack data structure
- Implementation of Stack
- Applications on Stack

Learning Topics

- The Queue data structure
- Implementing Queue
- Applications on Queue

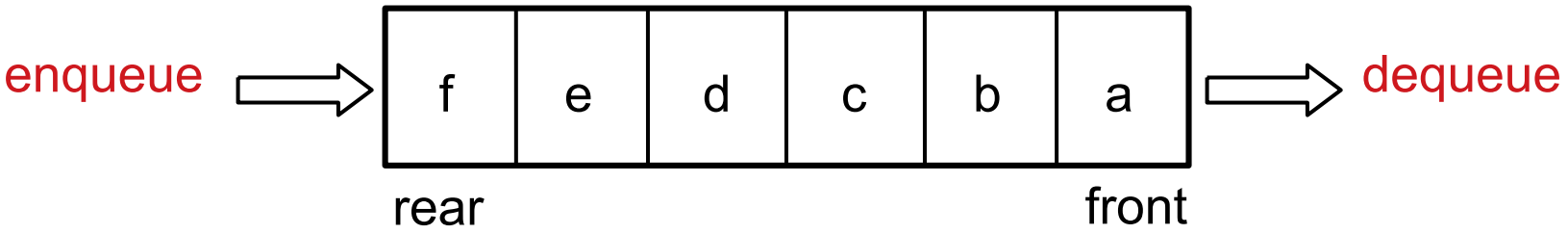
Queue

- Properties:
- FIFO (First In First Out)
- Many real-world situations: line to order food, …
- Access from both ends
  - Add to the rear
  - Remove from front
  - No immediate access to the middle elements



- Applications in resource management, printer, mail servers, packet transmission,...

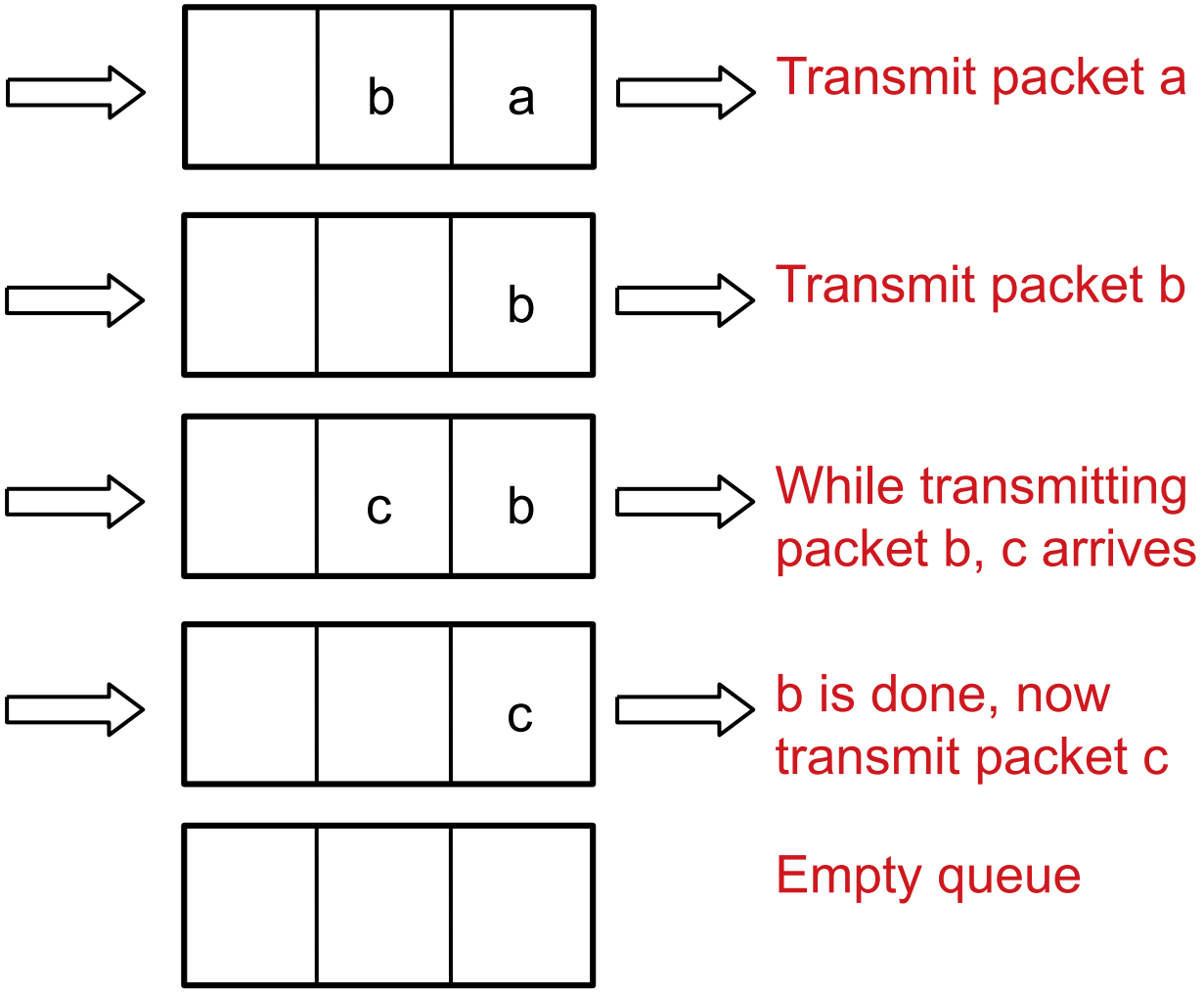
Queue



- Applications in resource management, printer, mail servers, packet transmission,...

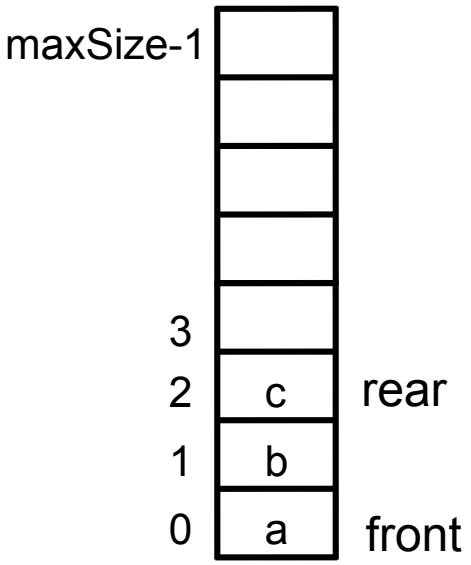
- Insert/enqueue
  - Adding an element to the rear of the queue
- Remove/dequeue
  - Removing and element from the front of the queue

Queue-Example



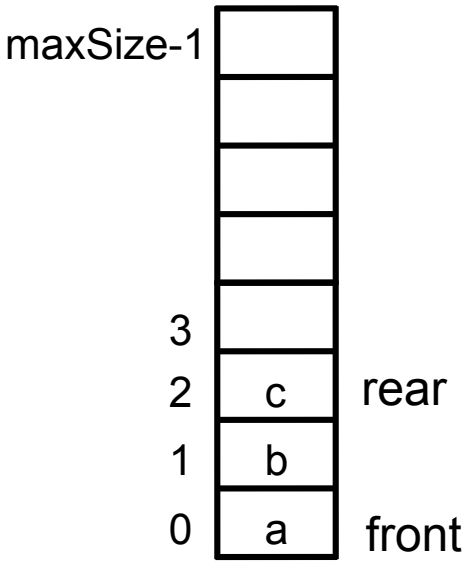
Queue-Implementation

- Can use an array to hold elements
- Fields needed
  - maxSize (capacity of queue)
  - queArray (array of elements in queue)
  - front (index of the front)
  - rear (index of the rear)
  - nItems (number of elements or size)



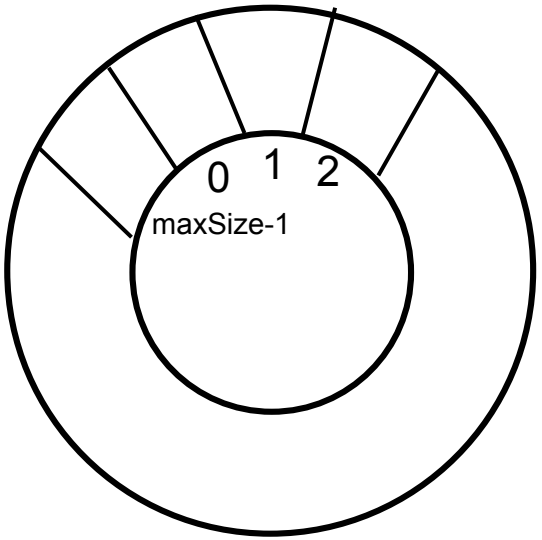
Queue-Implementation

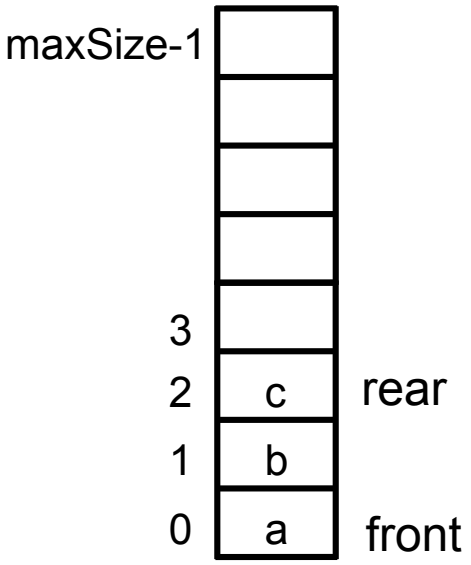
- Methods needed
  - enqueue(element)
  - dequeue()
  - peek()
  - size()
  - isEmpty()
  - isFull()



Queue-Implementation

- Rear and front move as we add/remove elements. What should we do if the rear reaches the maxSize-1?
  - Shift elements?
  - Circular Array





Queue Class

```
class Queue
{
        private int maxSize;
        private long[] queArray;
        private int front;
        private int rear;
        private int nItems;
//--------------------------------------------------------------
        public Queue(int s) // constructor
        {
                maxSize = s;
                queArray = new long[maxSize];
                front = 0;
                rear = -1;
                nItems = 0;
        }
```

Queue Class

```
public void enqueue(long j) // put item at rear of queue
        {
                if (isFull())
                        throw new IllegalStateException("Queue is full");
                if(rear == maxSize-1) // deal with wraparound
                        rear = -1;
                queArray[++rear] = j; // increment rear and insert
                nItems++; // one more item
        }

//--------------------------------------------------------------
public long dequeue() // take item from front of queue
        {
                if (isEmpty())
                        throw new IllegalStateException("Queue is empty");
                long temp = queArray[front++]; // get value and incr front
                if(front == maxSize) // deal with wraparound
                        front = 0;
                nItems--; // one less item
                return temp;
                }
```

Queue Class

```
public long peek() // peek at front of queue
{
        if (isEmpty())
                throw new IllegalStateException("Queue is empty");
        return queArray[front];
}
//-------------------------------------------------------------
public boolean isEmpty() // true if queue is empty
{
        return (nItems==0);
}
//-------------------------------------------------------------
public boolean isFull() // true if queue is full
{
        return (nItems==maxSize);
}
//-------------------------------------------------------------
public int size() // number of items in queue
{
        return nItems;
}
}
```

Queue Demo Class

```
class QueueDemo
{
        public static void main(String[] args)
        {
                Queue myqueue = new Queue(4);

                for(int i=1; i<=4; i++)
                        myqueue.enqueue(2*i);

                System.out.println(myqueue.dequeue());
                myqueue.enqueue(10);
                System.out.println(myqueue.dequeue());
                System.out.println(myqueue.peek());
                System.out.println(myqueue.size());
        }
}
```

Queue Class without nItems

```
public int size() // number of items in queue
{
        if(rear >= front) // contiguous sequence
                return rear-front+1;
        else // broken sequence
                return (maxSize-front) + (rear+1);
}
```

Applications on Queue

- Write a Queue client KthString.java that reads in a list of words, and an integer K, and prints the kth word from the end found on the input list of words, assuming that standard input has k or more strings.
- java KthString
  it was the best of times it was the worst of times
- 9
- best

More Applications on Stack

- Evaluation postfix expressions.

Infix Expression	Prefix Expression	Postfix Expression
A + B * C + D	+ + A * B C D	A B C * + D +
(A + B) * (C + D)	* + A B + C D	A B + C D + *
A * B + C * D	+ * A B * C D	A B * C D * +
A + B + C + D	+ + + A B C D	A B + C + D +